WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, April 26, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L25	l2 and L22	6
☐	L24	l12.ab. and L22	11
☐	L23	l12 and L22	32
☐	L22	l17 or l18 or l19 or l20 or L21	1721
☐	L21	710/266.ccls.	210
☐	L20	710/261.ccls.	155
☐	L19	710/18.ccls.	265
☐	L18	713/320.ccls.	517
☐	L17	713/100.ccls.	648
		DB=USPT; PLUR=YES; OP=OR	
☐	L16	US-6193422-B1.did.	1
☐	L15	US-6193422-B1.did.	1
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L14	L13.ab.	7
☐	L13	L12 same ((based or depend$4 or respons$4) near3 event)	19
☐	L12	(processor near5 mode near5 switch$4)	1823
☐	L11	(hidden near5 event near3 handler)	7
☐	L10	(event near2 (based or depend$4 or respons$4)) same ((switch$4 or modif$7 or chang$4 or alter$4) near5 execut$4 near3 mode)	5
☐	L9	(event near2 handler) same ((switch$4 or modif$7 or chang$4 or alter$4) near5 execut$4 near3 mode)	2
☐	L8	(event near2 handler) same (hidden near2 memory)	2
☐	L7	L6.ab.	3
☐	L6	(load$4 near5 (event near2 handler))	26
☐	L5	L4 same computer	11
☐	L4	L3 same processor	16
☐	L3	L2 same memory	70
☐	L2	((mother adj board) or (motherboard)) same firmware	131
☐	L1	6009520.pn.	2

END OF SEARCH HISTORY

First Hit Fwd Refs

☐ Generate Collection | Print

L14: Entry 1 of 7 File: USPT Jan 6, 2004

DOCUMENT-IDENTIFIER: US 6675269 B2
TITLE: Semiconductor device with memory controller that controls page mode access

Abstract Text (1):
A memory controller and data processor have their operation mode switched from the page-on mode for high-speed access to a same page to the page-off mode in response to consecutive events of access to different pages, so that the memory access is performed at a high speed and low power consumption.

L11: Entry 3 of 7 File: USPT Sep 21, 1999

DOCUMENT-IDENTIFIER: US 5956481 A
TITLE: Method and apparatus for protecting data files on a computer from virus infection

Brief Summary Text (10):
Although the patch can offer protection of a file for selected open file events, the separate implementation of the patch prevents this solution from handling all open file events. For a word processing program, such as the "WORD" program, one will appreciate that there exists a wide variety of ways to open a document or file. External open events, such as selecting the "Open" command from a pull-down menu or double-clicking on a file displayed on the desktop, can be trapped by an event handler of the patch to initiate virus protection. The patch, however, cannot successfully detect all external open file events. In addition, internal open file events cannot be trapped by an external event handler because they are hidden within the internal code layer of the executable program. These internal open file events do not generate an action that is readily recognizable by a separate program, such as the patch solution. If all external and internal open file events cannot be addressed by a separate protection program, then it is possible for a hacker to circumvent the patch protection by exploiting this hole in the protection perimeter. To address external and internal open file events, it is necessary to incorporate virus protection within the executable program itself. Thus, there is a further need for a system for protecting a file or document from virus infection by including protection within the executable program compatible with these files.

L11: Entry 4 of 7 File: USPT Aug 4, 1998

DOCUMENT-IDENTIFIER: US 5790122 A
** See image for Certificate of Correction **
TITLE: Method for providing maximum screen real estate in computer controlled display systems during software application navigation

Brief Summary Text (11):
Instead of viewing the manipulation of application navigation elements such as menu's, buttons, icons or other screen objects by the CPU, for example, as the only relevant event, the present invention treats the display of the navigation elements by the CPU as the relevant event. In turn, this separate and heretofore unknown navigation displaying event may have its own event handler which when executed by the CPU provides for the modification of the navigation displaying event properties. By employing this separate event type which keeps navigation elements hidden until the event handler activates the navigation displaying event thereby activating the navigation event, the present invention overcomes the disadvantages attendant to known computer controlled display systems utilizing navigation elements fixed on a computer screen. As such, the present invention provides the advantages of reducing screen clutter, enhancing the visual impact of the graphic or other data, and maximizing utilization of precious screen space, without losing the ability to navigate a software application.

L10: Entry 1 of 5 File: USPT Sep 23, 1997

DOCUMENT-IDENTIFIER: US 5671422 A
TITLE: Method and apparatus for switching between the modes of a processor

Detailed Description Text (44):
During subsequent protected mode processing (e.g., application 150), another SMM event occurs. In response to this SMM event, the processor suspends the execution of application 150, switches to the SMM mode, stores the current return information for resuming execution of application 150, and again begins processing SMM routine 170 as shown by line 4. As a result, SMM routine 170 has two sets of return information available. In response to an RTC interrupt, the processor suspends the execution of SMM routine 170 to execute RTC SMM service routine 190 as shown by line 5. RTC SMM service routine 190 causes the processor to return to the protected mode to execute return routine 130 using the predetermined return information and a resume command as shown by line 6. Return routine 130, for example, allows for the servicing of protected mode events using the protected mode's event handling mechanism. Return routine 130 sets an RTC SMM indication and concludes by causing an SMM event. In response to the SMM event, the processor begins executing SMM routine 170 again starting at its initial instruction as shown by line 7. Since the RTC SMM indication indicates the current SMM event is an RTC SMM event, SMM routine 170 causes the processor to resume processing the RTC SMM service routine 190 as shown by line 8. RTC SMM service routine 190 causes the processor to resume the suspended execution of SMM routine 170 using an interrupt return command as shown by line 9. SMM routine 170 concludes by causing the processor to resume execution of application 150 using the current return information and a resume command as shown by line 10.

L10: Entry 2 of 5 File: USPT Apr 28, 1992

DOCUMENT-IDENTIFIER: US 5109329 A
TITLE: Multiprocessing method and arrangement

CLAIMS:

1. A method for use in a multiprocessor system having a first and a second processor and a plurality of means each for pointing to a procedure to be executed in response to an occurrence of an event to which the means corresponds, wherein execution mode changes from a user mode for executing user program instructions to a privileged mode for executing operating system instructions in response to occurrence of any one of a plurality of events, the method comprising the steps of:

executing a process in the user execution mode in the first processor;

changing from the user execution mode to the privileged execution mode in the first processor, in response to the occurrence in the first processor of any one of the plurality of events;

executing in the privileged execution mode in the first processor a first procedure pointed to by first means of the plurality of pointing means, the first means corresponding in the first processor to at least the occurred one of the plurality of events; and

transferring the process from the first processor to the second processor for continued execution, by the execution of the first procedure, wherein execution of the procedure pointed to be the pointing means corresponding in the first processor to any one of the plurality of events transfers the process from the first processor to the second processor for continued execution.

First Hit Fwd Refs

Generate Collection Print

L5: Entry 1 of 11 File: USPT Mar 23, 2004

DOCUMENT-IDENTIFIER: US 6710819 B2
TITLE: Method and system for improved display filtering

Detailed Description Text (6):
DIP 120, in one embodiment, retrieves or receives display data 115 and/or DIP instructions 105 from memory 110. In one embodiment, DIP 120 is implemented as hardware or firmware. For example, DIP 120 could include a peripheral connect interface (PCI) card, DIP 120 could be located on a computers motherboard, or DIP 120 could be implemented as a digital signal processor (DSP) on a computer's central processing unit (CPU). In another embodiment, DIP 120 is implemented as software instructions. For example, DIP 120 could include a subset of a plurality of instructions of a graphics display program. In yet another embodiment, DIP 120 is implemented using a combination of software instructions and hardware or firmware.

L5: Entry 2 of 11 File: USPT Sep 9, 2003

DOCUMENT-IDENTIFIER: US 6618810 B1
**** See image for Certificate of Correction ****
TITLE: Bios based method to disable and re-enable computers

Detailed Description Text (4):
CPU 102 can be constructed from one or more microprocessors and/or integrated circuits. During operation, main memory 106 is loaded with programs and data that CPU 102 may access. When computer system 100 starts up, CPU 102 passes control to basic input/output system (BIOS) program 118, which is typically located in read only memory (ROM) 119 as part of computer system's 100 firmware. Traditionally, BIOS program 118 is implemented n Erasable Programmable Read Only Memory (EPROM). EPROM has an advantage of not being modified in circuit. To modify the contents of the EPROM, the device must be first erased by being removed from computer system 100 and exposed to ultraviolet light for a prolonged period of time. In this respect, BIOS is not easily upgradeable or replaceable once computer system 100 is assembled and placed in the field, thereby causing unnecessary cost and inconvenience of equipment downtime to customers. Because of the importance of field upgrading, at least a portion of all BIOS firmware is implemented using volatile ROM, known in the art as flash ROM, thereby allowing customers to upgrade and make changes to the BIOS program 118 using not only traditional off-board PROM programming equipment, but also on-board updates and in-system updates where the flash ROM remains physically attached to the motherboard (not shown) in computer system 100 during the update. On-board updates involve an external connection that supplies all signals and voltages required for programming/erasing the flash memory, with an external processor (outside of computer system 100) executing the update algorithm. In-system updates may be performed in several ways including running an update utility that is downloaded from a network or that is distributor on a mass storage device such as a hard disk, floppy disk, or CD-ROM that can be operably connected to computer system 100.

L5: Entry 3 of 11 File: USPT Jun 4, 2002

DOCUMENT-IDENTIFIER: US 6401202 B1
TITLE: Multitasking during BIOS boot-up

Brief Summary Text (5):
To execute a program, a processor in a computer system has to access the executable code of the program from the memory. When the computer is first powered up, the processor usually executes the firmware which generally is the Basic Input and Output System (BIOS) program. The BIOS contains the executable code for a boot-up program. It is essentially a built-in software containing a set of instructions that control systems devices and test memory. Because it is typically stored in a Read-Only Memory (ROM) chip on the motherboard, it is sometimes call the ROM BIOS.

L5: Entry 4 of 11 File: USPT Oct 19, 1999

DOCUMENT-IDENTIFIER: US 5968141 A
TITLE: System for selectively upgrading firmware code for optical disk drive via ATA/IDE interface based on host system programming enable signal

Detailed Description Text (15):
On the other hand, when the programming controller 32 connects the memory device 34 to the IDE bus 10 of the optical disk drive unit, the processor in the host computer system is then allowed to implement its access to the memory device 34. Both write and read accesses to the memory device 34 are allowed for facilitating the on-site firmware code upgrade in a series of read/write operations in the memory device. This is an operation similar to the on-board upgrading of the system BIOS (Basic Input/Output System) code for the IBM-compatible computer systems. In such on-board upgrading operations, the memory device where the BIOS code resides needs not be removed from the motherboard of the computer system, and no dedicated semiconductor memory programmer equipment is required. With the proper setting of a few switches, the code upgrade operation can be completed by the computer system executing a software program.

L5: Entry 6 of 11 File: USPT Mar 30, 1999

DOCUMENT-IDENTIFIER: US 5887837 A
TITLE: Pivotal computer stand

Detailed Description Text (34):
The count stored in counter 176 can also be read to determine the total amount of time that the computer 10 has been in a full-on state. The counter 176 is preferably read through a suitable Basic Input/Output Service (BIOS) call and can also be reset through a suitable BIOS call. Since one of ordinary skill in the art is capable of generating the suitable BIOS calls for reading and reading the count, the exact commands will not be described in full detail. Rather than BIOS calls, however, the count stored in counter 176 could instead be read through direct memory access. The life-time count may alternatively be realized in other ways than through the BIOS. For instance, a specialized ASIC may be designed to implement a timer and a counter. Also, a keyboard controller can be programmed to provide the timer and counter functions. The keyboard controller, which is typically found on most motherboards, has firmware which already is involved in power management and may therefore be easily modified to include the processor 172 and to track the life-time of the computer. Other ways of realizing the life-time counter will become apparent to those skilled in the art.

L5: Entry 10 of 11 File: USPT May 19, 1998

DOCUMENT-IDENTIFIER: US 5754852 A
TITLE: Apparatus for combining cellular telephone ring signals and PSTN ring signals

Brief Summary Text (6):
The Family I models typically used the popular INTEL 8088 or 8086 microprocessor as the system processor. These processors have the ability to address one megabyte of memory. Later Family I models and the Family II models typically use the higher speed INTEL 80286, 80386, and 80486 microprocessors which can operate in a real mode to emulate the slower speed INTEL 8086 microprocessor or a protected mode which extends the addressing range from 1 megabyte to 4 Gigabytes for some models. In essence, the real mode feature of the 80286, 80386, and 80486 processors provide hardware compatibility with software written for the 8086 and 8088 microprocessors. As personal computer technology has developed and moved from eight to thirty-two, and eventually sixty-four bit wide bus interaction and higher speed microprocessors capable of real and protected mode operation, performance capability has been sought by separating the architecture of the personal computer into varying bus areas. More specifically, in the original IBM PC, what came to be known as an expansion bus was essentially a direct extension of the microprocessor (8086 or 8088) connections, buffered and demultiplexed as required. Later, as the AT bus specification was developed and came into wide use (now being also known as the Industry Standard Architecture or ISA), it became possible to sever the nearly direct connection between the microprocessor and the bus, giving rise to the presence of what became known as the local processor bus and the renaming of the expansion bus as the input/output bus. Typically, in order to enhance performance, the local processor bus runs at a higher clock speed (typically expressed in Hertz) than does the input/output bus. The IBM AT architecture also opened the possibility of running more than one microprocessor on the input/output bus, through use of direct memory access (DMA) interrupts. Beginning with the earliest personal computer system of the Family I models, such as the IBM Personal Computer, it was recognized that software compatibility would be of utmost importance. In order to achieve this goal, an insulation layer of system resident code, also known as "firmware", was established between the hardware and software. This firmware provided an operational interface between a user's application program/operating system and the device to relieve the user of the concern about the characteristics of hardware devices. Eventually, the code developed into a BASIC input/output system (BIOS), for allowing new devices to be added to the system, while insulating the application program from the peculiarities of the hardware. The importance of BIOS was immediately evident because it freed a device driver from depending on specific device hardware characteristics while providing the device driver with an intermediate interface to the device. Since BIOS was an integral part of the system and controlled the movement of data in and out of the system processor, it was resident on the system planar and was shipped to the user in a read only memory (ROM). For example, BIOS in the original IBM Personal Computer occupied 8K of ROM resident on the planar board (motherboard).

L9: Entry 1 of 2 File: USPT Sep 18, 2001

DOCUMENT-IDENTIFIER: US 6292792 B1
TITLE: System and method for dynamic knowledge generation and distribution

CLAIMS:

19. The system of claim 8, wherein the network is the internet and the browser is an internet browser, and further wherein the event handler further comprises:

a pop-up handler executable to receive an input from the event handler and, if a pop-up up window is associated with the input, creates the pop-up window for viewing by the user;

a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, creates the mode change for viewing by the user; and

an action sender executable to send a request based on the input from the event handler to the tutor maker program.

27. The system of claim 26, wherein the event handler further comprises:

a pop-up handler executable to receive an input from the event handler and, if a pop-up up window is associated with the input, create the pop-up window for viewing by the user;

a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and

an action sender executable to send a request based on the input from the event handler to the tutor maker program.

49. The system of claim 48, wherein the user interface further comprises:

a browser that enables browsing of the network; and

a browser extension, comprising an event handler executable to allow the user to generate an event to be acted upon by the tutor maker program, wherein the event handler further comprises:

a pop-up handler executable to receive an input from the event handler and, if a pop-up up window is associated with the input, create the pop-up window for viewing by the user;

a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and

an action sender executable to send a request based on the input from the event handler to the tutor maker.

58. The system of claim 57, wherein the network is the internet and the browser is an internet browser, and further wherein the event handler further comprises:

a pop-up handler executable to receive an input from the event handler and, if a pop-up up window is associated with the input, create the pop-up window for viewing by the user;

a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and

an action sender executable to send a request based on the input from the event handler to the tutor maker.

L25: Entry 1 of 6 File: USPT Mar 9, 2004

DOCUMENT-IDENTIFIER: US 6704864 B1
TITLE: Automatic configuration of equipment software

Detailed Description Text (106):
As mentioned hereinabove in one alternative embodiment, the automatic configuration update process may be triggered by simply inserting the devices into the PC 302 and providing power, which subsequently initiates identification and interrogation of the devices to determine if the existing drivers need to be updated by downloading and installing of the latest firmware and driver software from the VWS 2504. The PC 302 also contains a non-volatile memory 3174 that contains the system BIOS. The CPU 3100 executes the system BIOS contained in the nonvolatile memory 3174 during power up of the PC 302, and such system BIOS can also be updated using the disclosed architecture and software described hereinabove for automatically detecting and updating the various components and cards associated with the PC 302. This process can be facilitated in a number of ways, for example, when the user obtains a motherboard (also called a system board) which includes the non-volatile memory 3174, the manufacturer of the motherboard can include an MRC label 3176 which could then be removed and placed on the outside of the computer chassis such that the user may periodically scan the MRC 3176 to facilitate or initiate the retrieval and installation of the latest software relevant to the motherboard BIOS stored in the non-volatile memory 3174. The motherboard MRC 3176 then contains a unique code associated with a model number of the particular motherboard and version number of the associated system BIOS. This information is then transmitted to in the same manner indicated hereinabove to retrieve the latest firmware updates or drivers required for the operating system to "recognize" the various features of the motherboard (e.g., ATA66 drive compatibility, high density floppy drives, etc.).

Current US Cross Reference Classification (1):
713/100